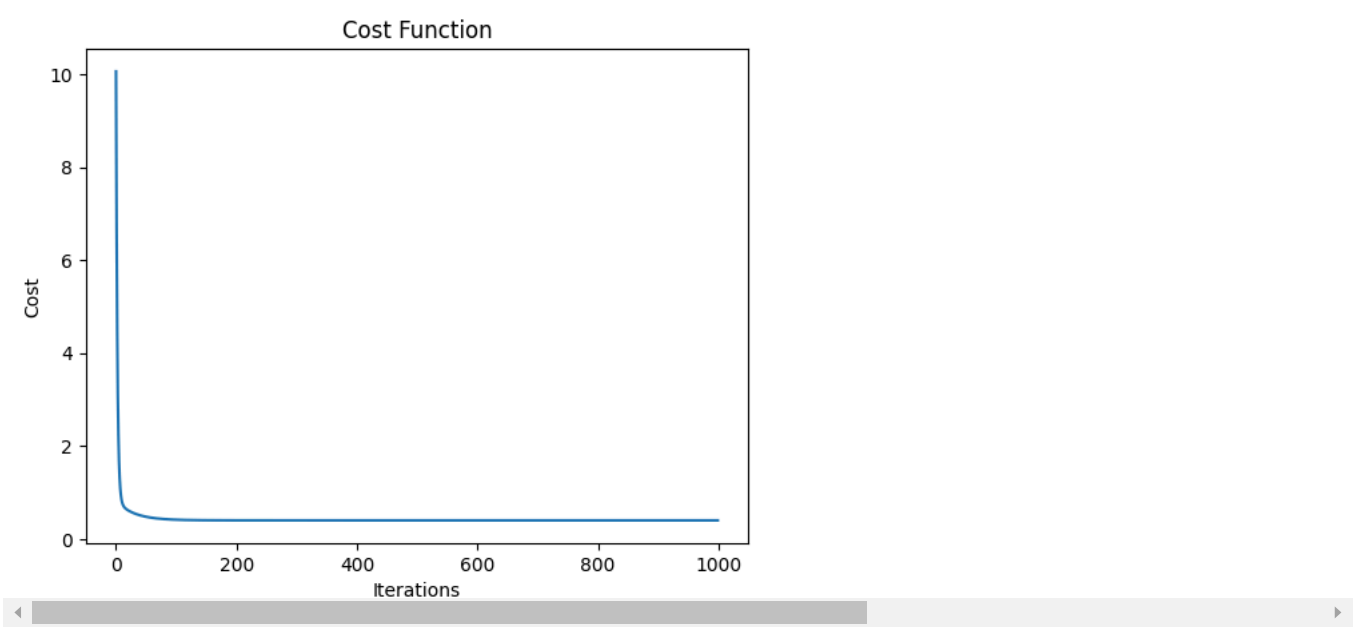
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)


X_b = np.c_[np.ones((100, 1)), X]


def compute_cost(X, y, theta):
    m = len(y)
    predictions = X.dot(theta)
    cost = (1/(2*m)) * np.sum(np.square(predictions - y))
    return cost


def gradient_descent(X, y, theta, alpha, iterations):
    m = len(y)
    cost_history = np.zeros(iterations)

    for i in range(iterations):
        predictions = X.dot(theta)
        errors = np.dot(X.transpose(), (predictions - y))
        theta -= (alpha/m) * errors
        cost_history[i] = compute_cost(X, y, theta)

    return theta, cost_history


theta = np.random.randn(2,1)

alpha = 0.1
iterations = 1000
theta, cost_history = gradient_descent(X_b, y, theta, alpha, iterations)

print("Theta:", theta)
print("Final cost:", cost_history[-1])
```

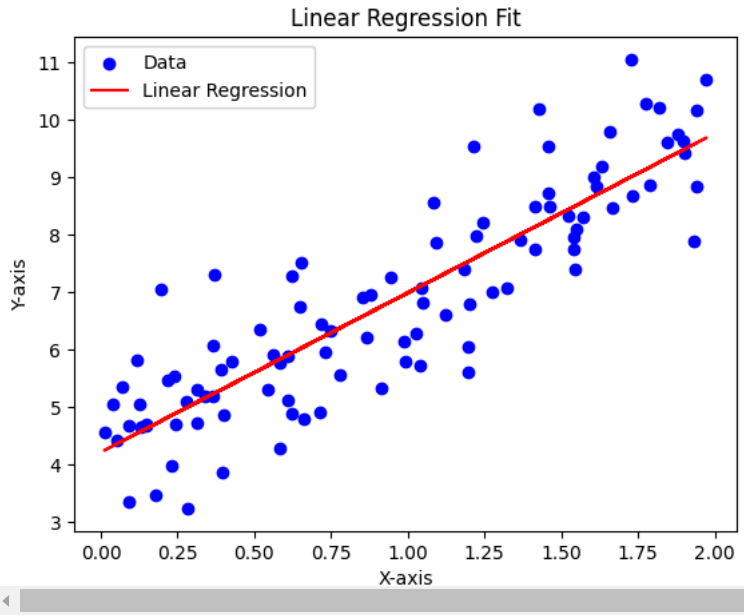
```
Theta: [[4.21509609]
 [2.77011344]]
Final cost: 0.4032922819835273
```

```
plt.plot(range(iterations), cost_history)
plt.xlabel("Iterations")
plt.ylabel("Cost")
plt.title("Cost Function")
plt.show()
```




```
plt.scatter(X, y, color='blue', label='Data')
plt.plot(X, X_b.dot(theta), color='red', label='Linear Regression')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Linear Regression Fit")
plt.legend()
plt.show()
```

```
plt.show()
```




```
import pandas as pd
ds= pd.read_csv("/content/housing.csv")
ds.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_va
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	45260
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	35850
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	35210
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	34130
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	34220

Next steps: Generate code with ds | View recommended plots | New interactive sheet

```
ds=ds.dropna()


X = ds.drop('median_house_value', axis=1)
y = ds['median_house_value']


X = pd.get_dummies(X, drop_first=True)


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
y_pred = model.predict(X_test)

from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R^2 Score:", r2)
```

```
Mean Squared Error: 4802173538.60416
R^2 Score: 0.6488402154431994
```

```
import pandas as pd
data=pd.read_csv("/content/HR.csv")
data.head()
```

	ID	Name	Department	GEO	Role	Rising_Star	Will_Relocate	Critical	Trending Perf	Talent_Level	...	salary	Gender
0	1	BRADDY	Operations	US	VP	3	0	1	8	6	...	low	M
1	2	BORST	Sales	UK	Senior Director	4	0	1	10	8	...	low	F
2	3	BIRDWELL	Finance	France	Senior Director	1	0	0	2	3	...	medium	F
3	4	BENT	Human Resources	China	Senior Director	4	0	1	8	7	...	high	M
4	5	BAZAN	IT	Korea	Director	4	0	1	8	7	...	low	F

5 rows × 30 columns

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = data.drop(['Department', 'GEO', 'Role', 'ID', 'Name'], axis=1)

salary_mapping = {'low': 0, 'medium': 1, 'high': 2}
data['salary'] = data['salary'].map(salary_mapping)

data = data.dropna()

X = data.select_dtypes(include=[float, int]).drop('salary', axis=1)
y = data['salary']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R^2 Score:", r2)
```

```
Mean Squared Error: 0.3840223282761343
R^2 Score: 0.14492463243664266
```